

# JAVA SWING

## JFC (Java Foundation Class) ve Swing

- ❑ Java dilinde iki grafik ortamı tanımlanmıştır. Bunlar **awt** ve **swing** kütüphaneleridir.
- ❑ Bir çok programı hazır olarak java kütüphanelerinden alacağız, bir kısmını da burada temel kalıplar olarak kullanılmak üzere hazırlanmıştır.
- ❑ Java grafik sistemini anlamak için anlamamız gereken ilk kavram koordinat sistemidir.
- ❑ Koordinat birimi pixel (ekran görüntü elemanı) Digital bilgisayarda ekran görüntüsü bir çok pixelin bir arada kullanılmasıyla oluşur.
- ❑ Her bir pixel'in rengi, parlaklığı gibi çeşitli fiziksel özellikleri değiştirilerek ekranda veyakullanılan grafik ortamında görüntü oluşur.
- ❑ Java'da ve diğer bilgisayar grafik sistemlerinde piksel koordinatları karteziyen koordinat sistemi gibi düzenlenmiştir, ancak y eksenini aşağı doğru yönelmiştir.
- ❑ Bunun temel sebebi bilgisayar grafiklerinin ilk defa printerlar kullanılarak oluşturulmuş olması ve satır satır ileriye doğru giden printerlarda geriye doğru hareket imkanı bulunmamasıydı.
- ❑ Günümüzde böyle çalışma gereksinimi olmamasına rağmen , bu tarihi sebepten dolayı grafik sistemi bu şekilde standartlaştırılmıştır.

KAYNAK 1. A. Kadir GÜNEYTEPE: Java Swing, Başlangıçtan İleri Düzeye,  
Her Yönüyle GUI Geliştirme Klavuzu

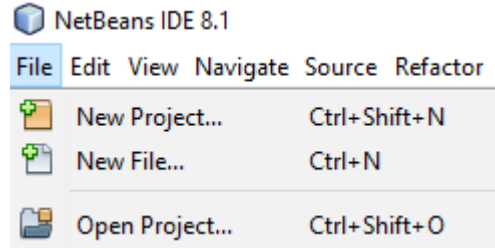
KAYNAK 2. M. TURHAN ÇOBAN: JAVA BİLGİSAYAR DİLİYLE PROGRAMLAMA

- ❑ Swing, yukarıda da belirtildiği gibi, Java için geliştirilmiş ve **JFC**'nin bir parçası olan, görsel bileşenler içeren ve grafiksel arayüze sahip uygulamalar geliştirmeyi sağlayan bir **API**'dir.
- ❑ Swing öncülü olan **AWT**'den (**Abstract Window Toolkit**) daha gelişmiş ve daha iyi donatılmış GUI bileşenleri içerir; ayrıca farklı platformlara kolayca uyum sağlayabilmesine izin veren **Eklenebilir Görünüm ve Davranış**'ı destekler.

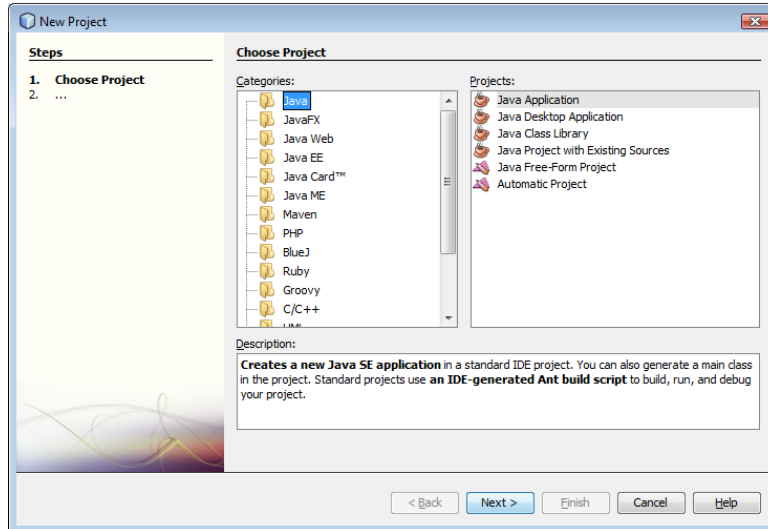
**JFC**, grafiksel kullanıcı arayüzü (GUI) geliştirmek için gerekli özellikler ve Java uygulamaları için zengin grafiksel işlevsellik ve etkileşim sağlayan sınıflar topluluğudur.

## Uygulama Oluşturma:

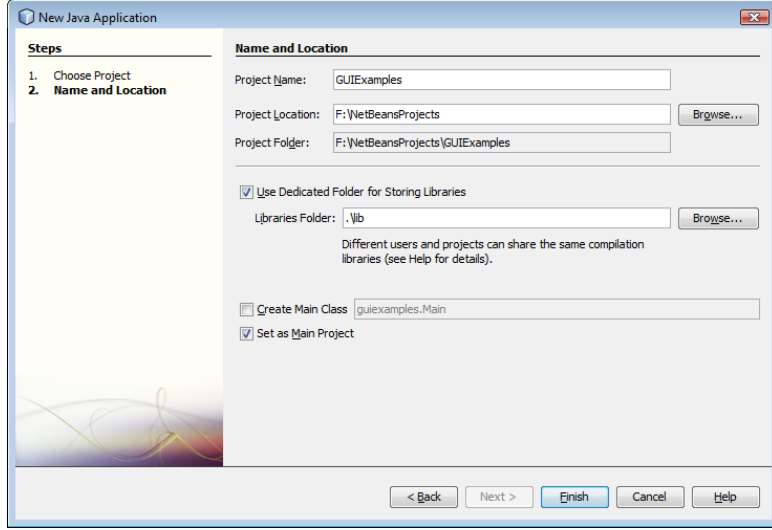
1. **NetBeans** üzerinde **File** menüsünden **New Project** seçilir.



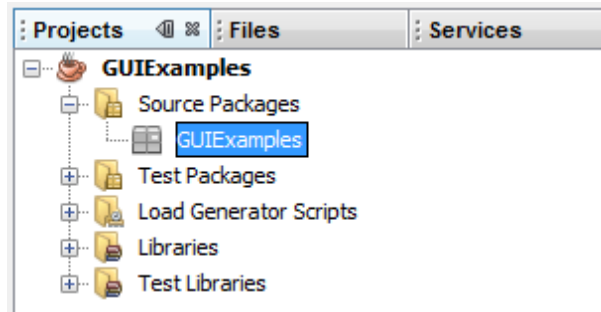
2. **Java Uygulamasını** Seçilir



### 3. Uygulama Adının Belirlenmesi

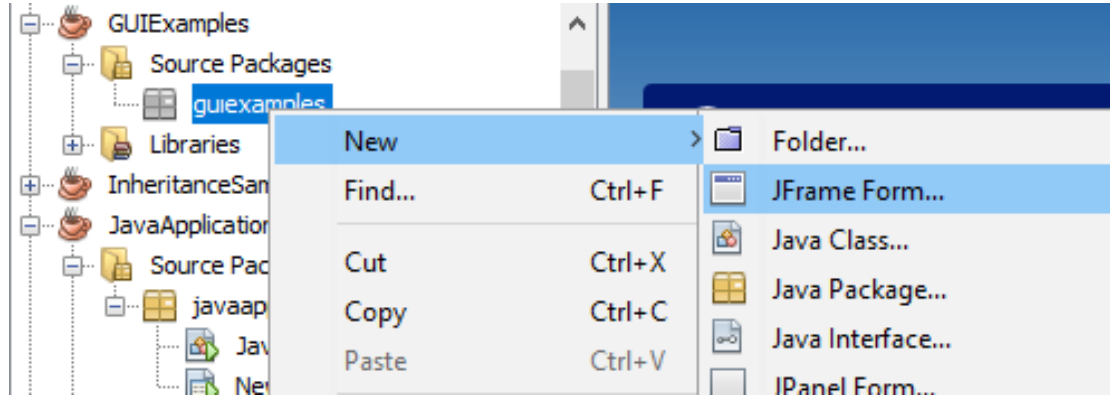


İşlem sonlandırıldığında, **NetBeans Projects** penceresinde aşağıdaki gibi bir görünüm oluşur.



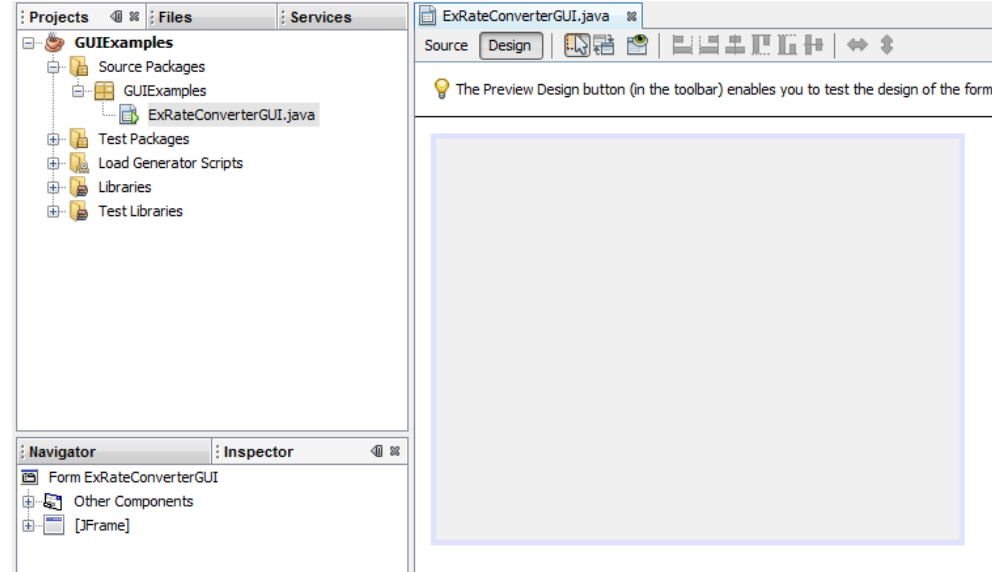
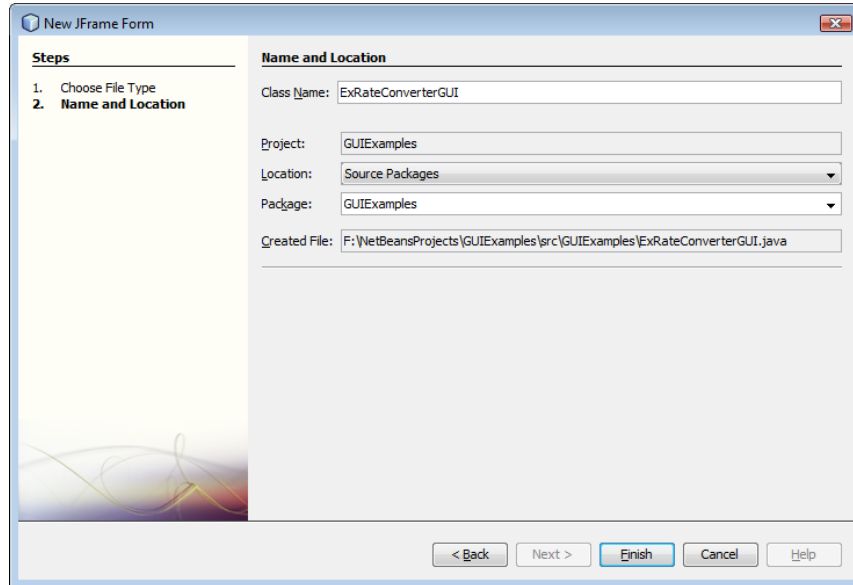
## 4. JFrame Form Ekleme

GUIExamples adlı proje üzerinde farenin sağ düğmesine tıklayarak, **New -> JFrame Form** seçilir.



## 5. GUI Sınıfını Adlandırma

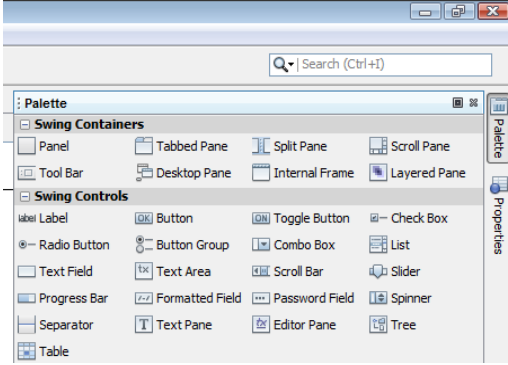
Name and Location alanında, **Class Name** bölümünde sınıf adı girilir. Sonuçta sağdaki gibi bir görünüm ortaya çıkar.



# NetBeans Arayüzü

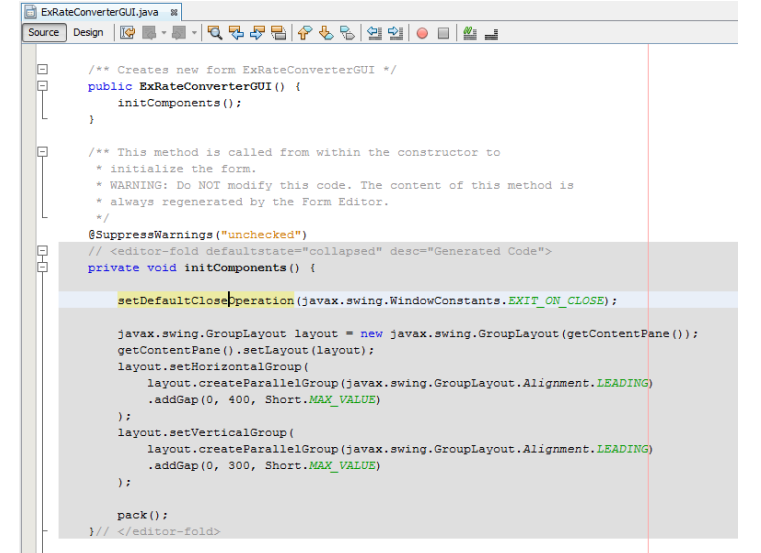
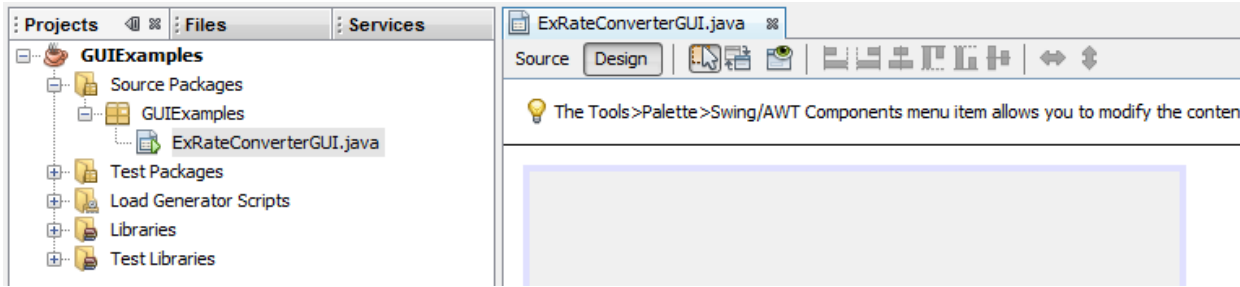
**NetBeans** geliştirme ortamının temel bazı bölümleri **Palette**, **Design Area**, **Property Editor** ve **Inspector** pencereleridir.

**Palette**, Swing API tarafından sağlanan bileşenleri içerir. **Palette** penceresinden, formun üzerine sürükleyip bırak yöntemiyle istenilen bileşen yerleştirilebilir.



## Design Alanı

**Design Area**, uygulamamızın görsel olarak oluşturulacağı alandır. **Source** ve **Design** olmak üzere iki bölümden oluşur. Bu iki bölüm arasında kendi adlarını taşıyan geçişli düğmelere basarak geçiş yapılabilir.

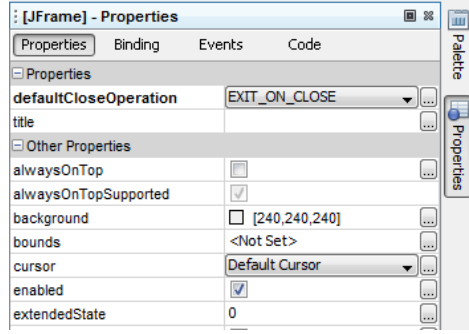


# Property Editor

**Property Editor**, forma eklediğimiz tüm bileşenlerin özelliklerini düzenleyebilmemizi sağlar.

Değiştirmek istenen özelliğe tıklanıp bu özellik rahatlıkla düzenlenebilir.

Böylece kod eklenmeden istenilen bir bileşenin özelliği kolayca değiştirilebilir. Aşağıda JFrame nesnesi seçili iken **Property Editor**'un görünümü verilmiştir.

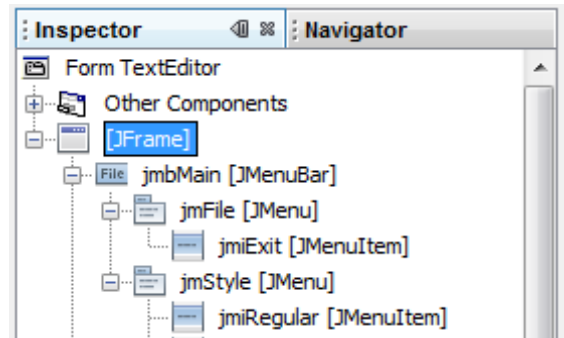


# Inspector

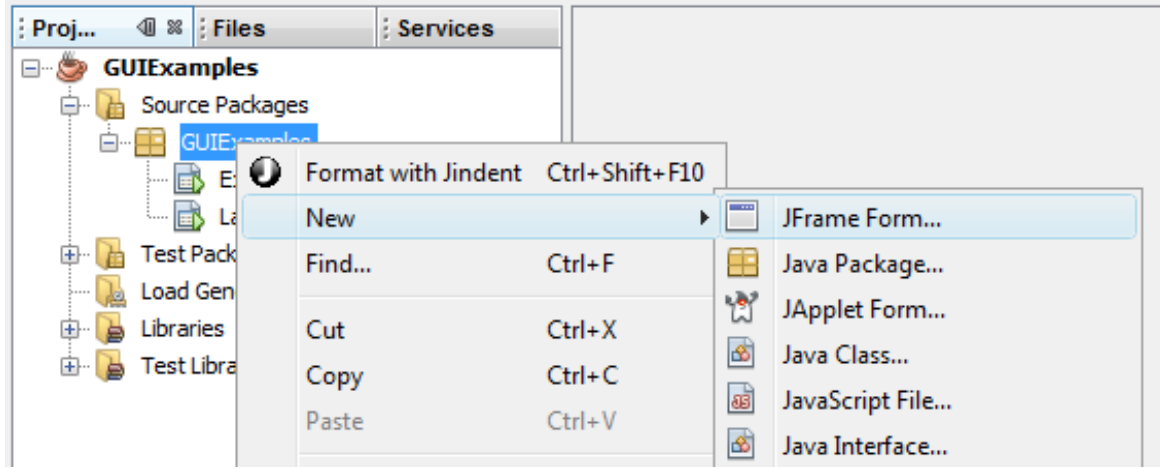
**Inspector**, uygulamamızda kullandığımız bileşenlerin görsel sunumunu sağlar.

Bileşen sıradüzeni buradan izlenebilir. Bu bölüm aynı zamanda bileşen adlarını değiştirmek için de kullanılabilir.

Aşağıda bir proje bileşen sıradüzenini gösteren bir **Inspector** penceresi görünümü verilmiştir.

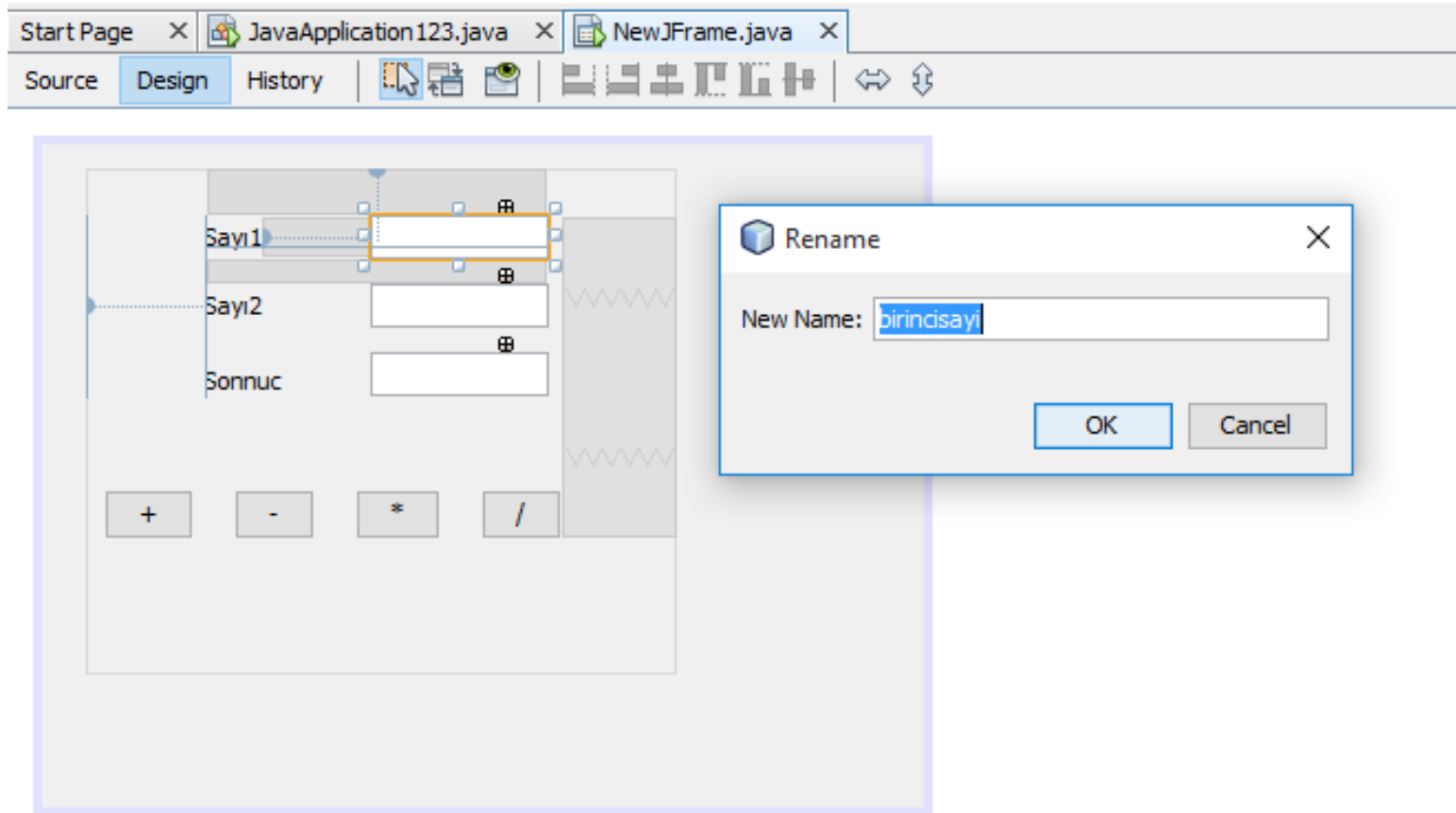


**Source Packages** üzerinde sađ tıkla açılan menüden, **JFrame Form** seçeneđi seçebilir.





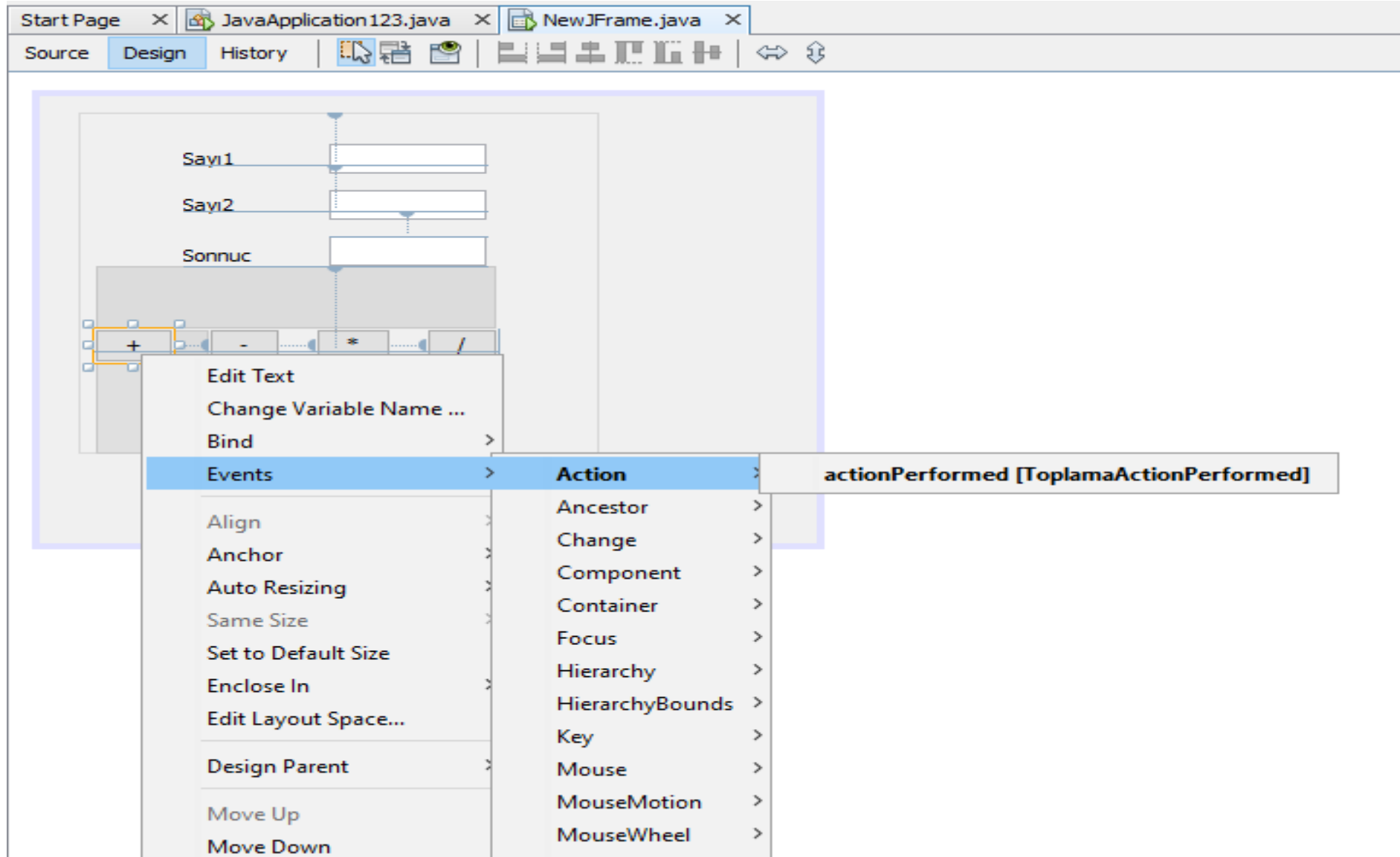
## 2. Adlandırma



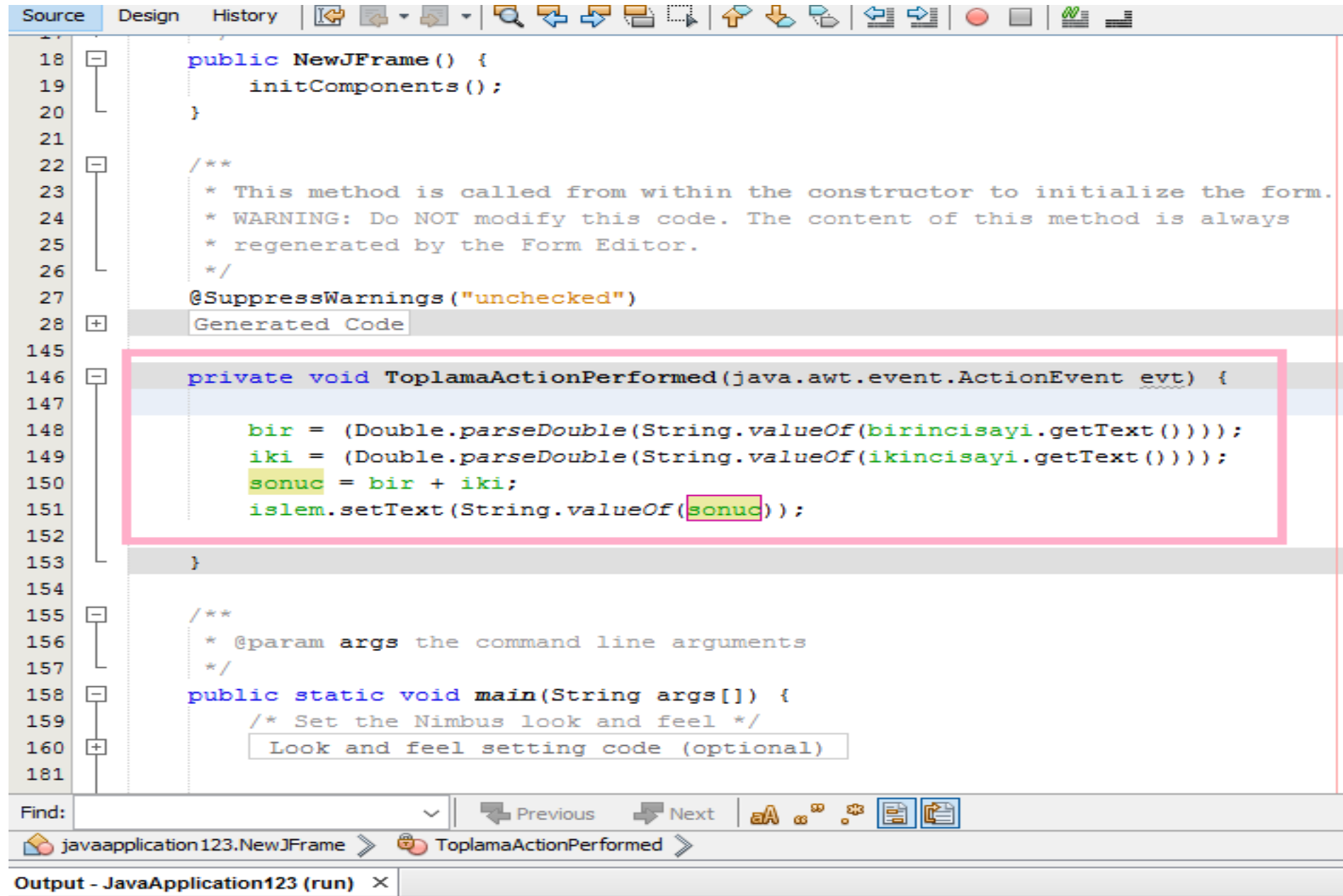
# 3. Değişken tanımlama

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package javaapplication123;
7
8  /**
9  *
10 * @author Caner Koç
11 */
12 public class NewJFrame extends javax.swing.JFrame {
13
14     double bir, iki, sonuc;
15     /**
16     * Creates new form NewJFrame
17     */
18     public NewJFrame() {
19         initComponents();
20     }
21
22     /**
23     * This method is called from within the constructor to initialize the form.
24     * WARNING: Do NOT modify this code. The content of this method is always
25     * regenerated by the Form Editor.
26     */
27     @SuppressWarnings("unchecked")
28     Generated Code
```

# 4. Komuta görev atama



# 5. Komut yazma



```
18 public NewJFrame () {
19     initComponents ();
20 }
21
22 /**
23  * This method is called from within the constructor to initialize the form.
24  * WARNING: Do NOT modify this code. The content of this method is always
25  * regenerated by the Form Editor.
26  */
27 @SuppressWarnings ("unchecked")
28 Generated Code
145
146 private void ToplamaActionPerformed (java.awt.event.ActionEvent evt) {
147
148     bir = (Double.parseDouble (String.valueOf (birincisayi.getText ()))) ;
149     iki = (Double.parseDouble (String.valueOf (ikincisayi.getText ()))) ;
150     sonuc = bir + iki ;
151     islem.setText (String.valueOf (sonuc)) ;
152 }
153
154
155 /**
156  * @param args the command line arguments
157  */
158 public static void main (String args []) {
159     /* Set the Nimbus look and feel */
160     Look and feel setting code (optional)
161 }
```

Find:

Previous Next

javaapplication123.NewJFrame > ToplamaActionPerformed >

Output - JavaApplication123 (run) x


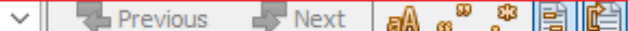
# 6. İşlem yapma

The image shows an IDE window with two tabs: 'JavaApplication123.java' and 'NewJFrame.java'. The code in 'NewJFrame.java' is as follows:

```
18 public NewJFrame () {
19     initComponents ();
20 }
21
22 /**
23  * This method is called from within the constructor to initialize the form.
24  * WARNING: Do NOT modify this code. The content of this method is always
25  * regenerated by the Form Editor.
26  */
27 @SuppressWarnings ("unchecked")
28 Generated Code
29
30 private void ToplamaActionPerformed (
31     java.awt.event.ActionEvent evt) {
32     bir = (Double.parseDouble (String
33     iki = (Double.parseDouble (String
34     sonuc = bir + iki;
35     islem.setText (String.valueOf (son
36 }
37
38 /**
39  * @param args the command line argu
40  */
41 public static void main (String args [
42     /* Set the Nimbus look and feel
43     Look and feel setting code (opt
```

Overlaid on the code is a window titled 'NewJFrame'. It contains three text input fields: 'Sayı1' with the value '20', 'Sayı2' with the value '42', and 'Sonuc' with the value '62.0'. Below the fields are four buttons: '+', '-', '\*', and '/'. The window has a standard title bar with minimize, maximize, and close buttons.

# 7. İşlemi çoğaltma

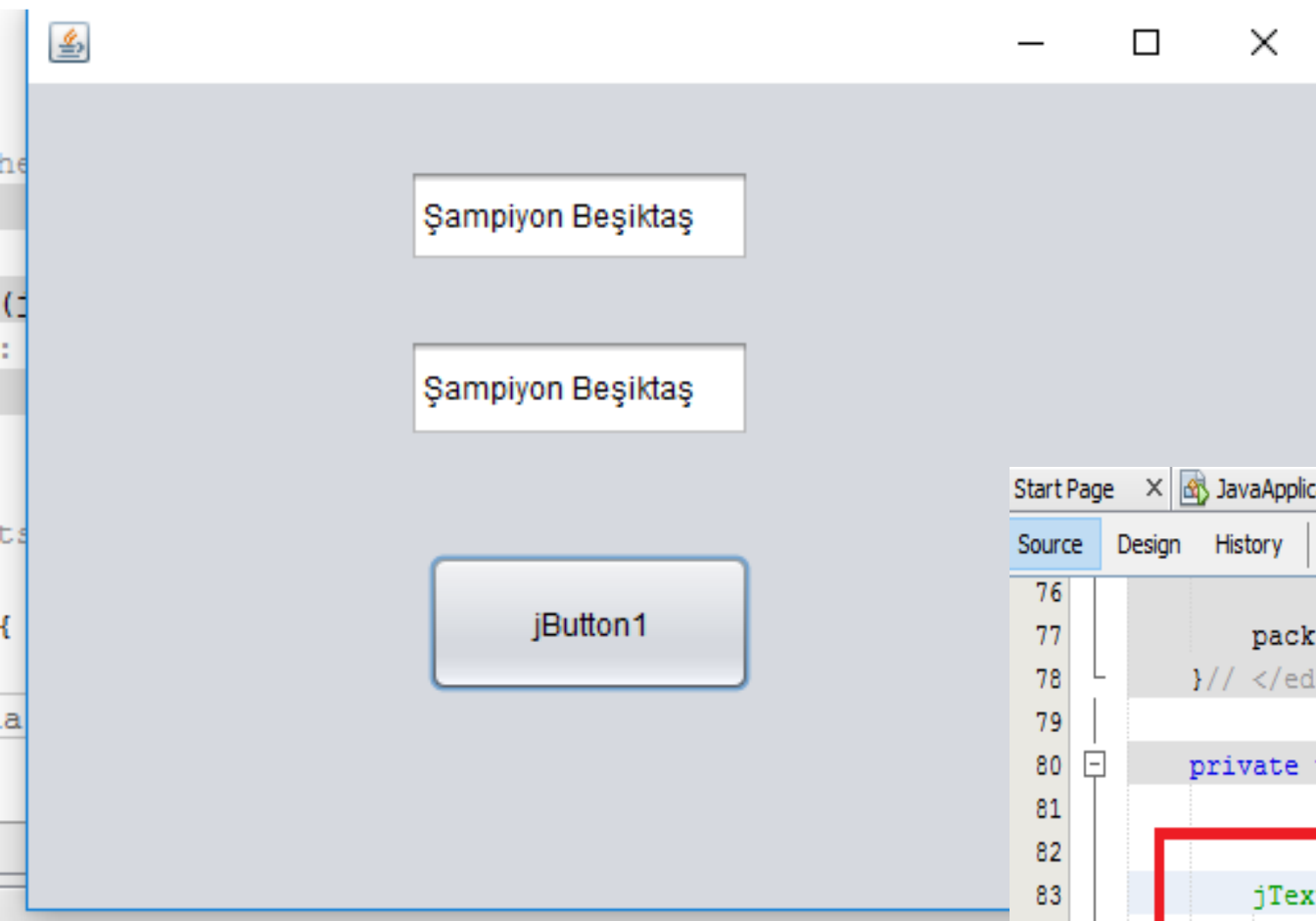
```
Source Design History 
161 private void ToplamaActionPerformed(java.awt.event.ActionEvent evt) {
162     bir = (Double.parseDouble(String.valueOf(birincisayi.getText())));
163     iki = (Double.parseDouble(String.valueOf(ikincisayi.getText())));
164     sonuc = bir + iki;
165     islem.setText(String.valueOf(sonuc));
166 }
167
168
169 private void CikarmaActionPerformed(java.awt.event.ActionEvent evt) {
170     bir = (Double.parseDouble(String.valueOf(birincisayi.getText())));
171     iki = (Double.parseDouble(String.valueOf(ikincisayi.getText())));
172     sonuc = bir - iki;
173     islem.setText(String.valueOf(sonuc)); // TODO add your handling code here:
174 }
175
176 private void CarpmaActionPerformed(java.awt.event.ActionEvent evt) {
177     bir = (Double.parseDouble(String.valueOf(birincisayi.getText())));
178     iki = (Double.parseDouble(String.valueOf(ikincisayi.getText())));
179     sonuc = bir * iki;
180     islem.setText(String.valueOf(sonuc)); // TODO add your handling code here:
181 }
182
183
184 private void bolmeActionPerformed(java.awt.event.ActionEvent evt) {
185     bir = (Double.parseDouble(String.valueOf(birincisayi.getText())));
186     iki = (Double.parseDouble(String.valueOf(ikincisayi.getText())));
187     sonuc = bir / iki;
188 }
Find: 
```

# ÖRNEK 1.

The image displays an IDE interface with two windows. The top window is in Design view, showing a graphical user interface with two empty text fields and a button labeled "jButton1". The bottom window is in Source view, showing the following Java code:

```
76  
77     pack();  
78     }  
79  
80     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
81  
82         jTextField1.getText();  
83         jTextField2.setText("null");  
84  
85  
86  
87     }  
88  
89     // TODO: add your handling code here:  
90  
91     java.awt.event.ActionEvent evt) {
```

A red rectangular box highlights the code on lines 82 and 83: `jTextField1.getText();` and `jTextField2.setText("null");`. Below the IDE, a small preview window shows the rendered application. It features two text fields; the top one is empty, and the bottom one contains the text "null". A button labeled "jButton1" is positioned below the text fields.



`jTextField1.getText();`

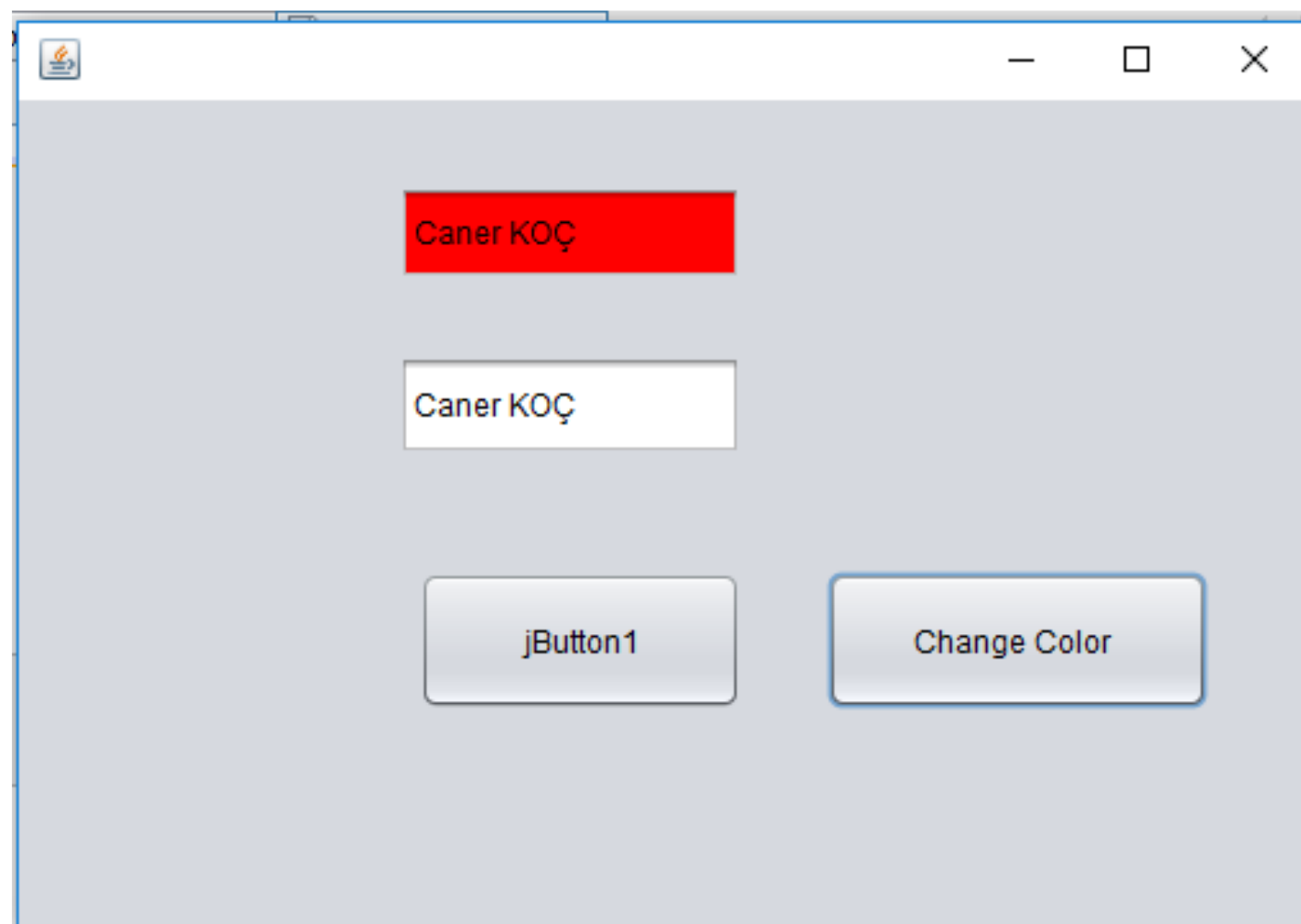
`jTextField2.setText();`

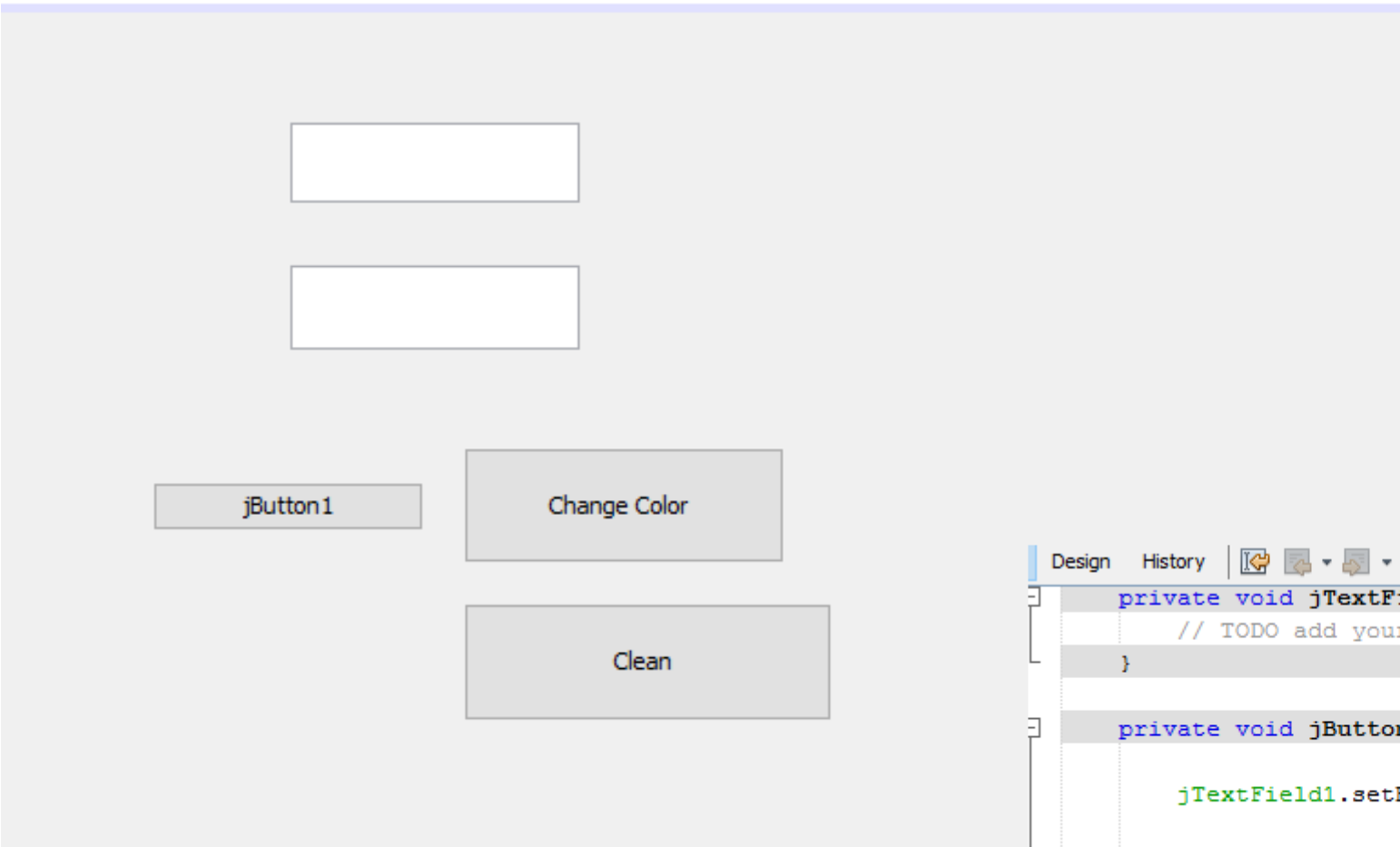
```
Start Page x JavaApplication128.java x NewJFrame.java x JavaApplication129.java x NewJFrame.java x
Source Design History
76
77     pack();
78 }// </editor-fold>
79
80 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
81
82
83     jTextField2.setText( jTextField1.getText());
84
85
86
87     // TODO add your handling code here:
88 }
89
90 private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
91     // TODO add your handling code here:
```



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setBackground(Color.red);  
}  
}
```

/\*\*





```
Design History | [Icons] | private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setBackground(Color.red);  
  
    // TODO add your handling code here:  
}  
  
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    jTextField1.setText(null);  
    jTextField2.setText(null);  
// TODO add your handling code here:  
}
```

# Mouse Görevleri

The screenshot displays an IDE interface with a Java Swing window titled "JavaApplication1 (run)". The window contains several UI components: a blue text field with "sdfgsdfgsd", a white text field with "sdfgsdfgsd", a button labeled "MauseEntered", a button labeled "jButton1", a "Change Color" button, and a "Clean" button. A context menu is open over the "jButton1" button, showing options like "Edit Text", "Events", "Align", "Anchor", etc. The "Events" menu is expanded, showing "Action", "Ancestor", "Change", "Component", "Container", "Focus", "Hierarchy", "HierarchyBounds", "Key", "Mouse", "MouseMotion", "MouseWheel", "PropertyChange", "VetoableChange", "InputMethod", and "Item". The "Mouse" sub-menu is also expanded, listing "mouseClicked", "mouseenter", "mouseExited", "mousePressed", and "mouseReleased". A mouse cursor is positioned over the "mouseenter" option. In the bottom right corner of the IDE, the page number "128 | 33" is visible.

```
private void jButton4MouseEntered(java.awt.event.MouseEvent evt) {
```

```
    jTextField1.setBackground(Color.yellow);
```

```
    // TODO add your handling code here:
```

```
}
```

```
private void jButton4MouseExited(java.awt.event.MouseEvent evt) {
```

```
jTextField1.setBackground(Color.black);
```

```
jTextField1.setBackground(Color.white);
```

```
// TODO add your handling code here:
```

```
}
```

